

Tips For Writing Portable C

Copyright 1999-2006, United States Government as represented by the Administrator of the National Aeronautics and Space Administration. No copyright is claimed in the United States under Title 17, U.S. Code.

This software and documentation are controlled exports and may only be released to U.S. Citizens and appropriate Permanent Residents in the United States. If you have any questions with respect to this constraint contact the GSFC center export administrator, <Thomas.R.Weisz@nasa.gov>.

This product contains software from the Integrated Test and Operations System (ITOS), a satellite ground data system developed at the Goddard Space Flight Center in Greenbelt MD. See <<http://itos.gsfc.nasa.gov>> or e-mail <itos@itos.gsfc.nasa.gov> for additional information.

You may use this software for any purpose provided you agree to the following terms and conditions:

1. Redistributions of source code must retain the above copyright notice and this list of conditions.
2. Redistributions in binary form must reproduce the above copyright notice and this list of conditions in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

This product contains software from the Integrated Test and Operations System (ITOS), a satellite ground data system developed at the Goddard Space Flight Center in Greenbelt MD.

This software is provided "as is" without any warranty of any kind, either express, implied, or statutory, including, but not limited to, any warranty that the software will conform to specification, any implied warranties of merchantability, fitness for a particular purpose, and freedom from infringement and any warranty that the documentation will conform to their program or will be error free.

In no event shall NASA be liable for any damages, including, but not limited to, direct, indirect, special or consequential damages, arising out of, resulting from, or in any way connected with this software, whether or not based upon warranty, contract, tort, or otherwise, whether or not injury was sustained by persons or property or otherwise, and whether or not loss was sustained from or arose out of the results of, or use of, their software or services provided hereunder.

1 Introduction

This document discusses portability issues encountered while porting the SMEX I&T TCW software from SunOS to Solaris, FreeBSD, and HP/UX.

Many of the issues discussed in this document are resolved by defining (or undefining) names and using “`#if ... #else ... #endif`” blocks to test whether or not those names are defined.

Currently these names get defined in the Makefiles via *GNU autoconf*.

This document doesn’t discuss the most important portability issues, which are good design and good coding style.

This document is on the WWW at <http://sunland.gsfc.nasa.gov/~tcw/portable/Top.html>.

2 Portability issues

2.1 bcopy() vs memcpy() (and bzero() vs memset())

Problem:

The functions “`bcopy()`” and “`bzero()`” are obsolete and may be missing in System V derived unices. “`memcpy()`” and “`memset()`” are available on all unices we are considering porting to.

Solution:

Replace “`bcopy(src,dest,len)`” with “`memcpy(dest,src,len)`”.
 Replace “`bzero(buf,len)`” with “`memset(buf,0,len)`”.

2.2 clock_gettime() vs gettimeofday()

Problem:

The preferred function for getting system time with sub-second precision is `clock_gettime()`, which is available on **Solaris 2.4**, but not on **SunOS 4.1.3**. The latter system provides the standard **Berkeley** function `gettimeofday()`.

Solution:

```
UNIX_TIME *ut;                                /* typedef in "unix_time.h" */
#ifndef HAVE_CLOCK_GETTIME
    struct timespec tp;
    rs = clock_gettime(CLOCK_REALTIME, &tp);
    ut->usec = (tp.tv_nsec + 500) / 1000;
#else
    struct timeval tp;
    rs = gettimeofday(&tp, NULL);
    ut->usec = tp.tv_usec;
#endif
    ut->sec = tp.tv_sec;
```

2.3 finite()

Problem: **FreeBSD 2.0.5**, **HPUX A.09.05**, **OSF1**, and **SunOS 4.1.3** don't have `ieeefp.h`; they declare

`extern int finite(double);`
 in `math.h`.

Solaris 2.4 declares

`extern int finite(double);`
 in `ieeefp.h` (and not in `math.h`).

Additionally, `finite()` might not be available at all on some HPUX configurations.

Solution:

```
#include <math.h>
#if defined(HAVE_IEEEFP_H)
#include <ieeefp.h>
#endif
```

Also, wrap *finit()* usages in #ifdef HASFINITE wrappers.

2.4 key_t

Problem:

FreeBSD 2.0.5 defines *key_t* in *sys/ ipc.h* instead of the more usual *sys/types.h*. (**HPUX A.09.05.5**, **OSF1**, **Solaris 2.4**, and **SunOS 4.1.3** all define *key_t* in *sys/types.h*).

Solution:

```
#include <sys/types.h>
#include <sys/ ipc.h>
```

2.5 malloc.h

Problem In **FreeBSD 2.0.5**, *malloc.h* is obsolete; *stdlib.h* should be used instead.

Solution:

```
#include <stdlib.h>
#if defined(HAVE_MALLOC_H)
#include <malloc.h>
#endif
```

2.6 select() and ulimit() vs. FD_SETSIZE

Problem:

SunOS suggested using the value returned by “*ulimit()*” as the first arg to “*select()*”. **FreeBSD** does not have “*ulimit()*”. Instead, **FreeBSD** recommends using “*FD_SETSIZE*”.

Solution:

Use “*FD_SETSIZE*” as the first arg to “*select()*”.

2.7 semctl() and union semun

Problem:

“*semctl()*” is declared with four arguments in **FreeBSD** but takes either three or four arguments (depending on the third) in other unixes. The fourth argument

is either “union *semun*” (on **SunOS 4.1.3**, **Solaris 2.4**, **OSF/1** and **FreeBSD**) or one of *int*, *unsigned short* *, etc. in **HP-UX 9.05**

The **SunOS 4.1** document *Porting Software to SPARC* says:

Programs that call *semctl()* with these subcommands [those requiring a fourth argument] must pass the union itself, rather than an element of the union, or a constant such as 0 (zero). Programs that call *semctl()* with other subcommands should omit the fourth argument, rather than pass a constant such as 0 (zero).

Fortunately, Sun technical support says it’s OK to pass a “union *semun*” fourth argument in all cases. (It is **NOT OK** to pass a constant such as 0 when the fourth argument is not needed).

To further complicate matters, **SunOS 4.1.3** and **FreeBSD 2.0.5**, define *union semun* in ‘*sys/sem.h*’. On other systems using *union semun*, programmers must define the union in any program using it.

Solution:

```
#ifdef NEED_UNION_SEMUN
union semun
{
    int val;
    struct semid_ds *buf;
    ushort *array;
};

#ifndef USES_UNION_SEMUN
#define USES_UNION_SEMUN
#endif
#endif /* NEED_UNION_SEMUN */

#ifndef USES_UNION_SEMUN
#define DeclareSemctlArg      union semun semun_arg
#define SemctlArrayArg(v)      (semun_arg.array = (v), semun_arg)
#define SemctlStructArg(v)    (semun_arg.buf = (v), semun_arg)
#define SemctlIntArg(v)       (semun_arg.val = (v), semun_arg)
#else
#define DeclareSemctlArg      /* nothing ... must be final decl. in list !
#define SemctlArrayArg(v)      (v)
#define SemctlStructArg(v)    (v)
#define SemctlIntArg(v)       (v)
#endif /* USES_UNION_SEMUN */

.
.
.

{ /* This is a portable semctl call */
    int unused;
    DeclareSemctlArg;
    val = semctl(semid, semnum, GETVAL, SemctlIntArg(unused));
}
```

```

.
.
.

{ /* This is a portable semctl call */
    DeclareSemctlArg;
    rc = semctl(semid, semnum, SETALL, SemctlArrayArg(vals));
}

```

2.8 statfs vs statvfs

Problem:

A few systems use `statfs()` to provide information about mounted file systems, other systems provide `statvfs()`. The .h files used with these functions vary between systems. The `statvfs()` function is defined by the Single UNIX Specification, Version 3, and all major branded UNIX operating systems, so that is the preferred function. We know of no cases where the `statvfs()` function is provided but ‘`sys/statvfs.h`’ is missing.

FreeBSD 4.4

Provides `statfs()` and uses ‘`sys/param.h`’ and ‘`sys/mount.h`’.
FreeBSD does not have `sys/vfs.h`.

Solaris Provides `statvfs()` and uses `sys/statvfs.h`.

HPUX 11

IRIX 6.5

Tru64 5.1B

Red Hat Linux 7.2

Red Hat Enterprise v.3

Provides both `statfs()` and `statvfs()`. Uses ‘`sys/vfs.h`’ with `statfs()` and ‘`sys/statvfs.h`’ with `statvfs()`.

AIX 5.2 Provides both `statfs()` and `statvfs()`. Uses ‘`sys/statfs.h`’ with `statfs()` and ‘`sys/statvfs.h`’ with `statvfs()`.

Solution:

The ITOS `configure` tests for the presence of `statvfs()`, and for ‘`sys/statvfs.h`’ and ‘`sys/vfs.h`’.

```

#ifndef HAVE_STATVFS /* statvfs() is preferred */
# include <sys/statvfs.h>
# define FSTATVFS fstatvfs
# define STRUCT_STATVFS struct statvfs
# define F_FRSIZE f_frsize /* fundamental filesystem block
                           size element in STRUCT_STATVFS */
#else
# include <sys/param.h> /* FreeBSD */
# include <sys/mount.h> /*      "      */
# ifdef HAVE_SYS_VFS_H

```

```

#      include <sys/vfs.h> /* Linux   */
# endif
# define FSTATVFS fstatfs
# define STRUCT_STATVFS struct statfs
# define F_FRSIZE f_bsize /* fundamental filesystem block
                           size element in STRUCT_STATVFS  */
#endif

...
STRUCT_STATVFS sfs;
...
rc = STATVFS(filename, &sfs);
blocksize = sfs.F_FRSIZE;

```

2.9 sys_errlist and sys_nerr

Problem

sys_errlist and *sys_nerr* are declared differently on different systems.

But that's OK – it turns out they're not needed anyway:

Solution

Don't use *sys_errlist* or *sys_nerr* – use *strerror()* instead!

```

#include <stdio.h>
#include <string.h>

extern int errno;
.
.
.
printf("system error message is \"%s\"\n", strerror(errno));

```

2.10 sys/socket.h

Problem:

FreeBSD 2.0.5, HPUX A.09.05.5, OSF1, and Solaris 2.5 all declare *accept()*, *bind()*, *connect()*, *getpeername()*, *getsockname()*, *getsockopt()*, *listen()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendto()*, *sendmsg()*, *setsockopt()*, *shutdown()*, *socket()*, and *socketpair()* in *sys/socket.h*.

SunOS 4.1.3 doesn't declare these functions in any of its standard header files.

Solution:

```

#include <sys/socket.h>
#ifndef DECLARED_SOCKET_FUNCS
extern int accept(int, struct sockaddr *, int *);
extern int bind(int, struct sockaddr *, int);

```

```

extern int connect(int, struct sockaddr *, int);
extern int getpeername(int, struct sockaddr *, int *);
extern int getsockname(int, struct sockaddr *, int *);
extern int getsockopt(int, int, int, void *, int *);
extern int listen(int, int);
extern int recv(int, void *, size_t, int);
extern int recvfrom(int, void *, size_t, int, struct sockaddr *, int *);
extern int recvmsg(int, struct msghdr *, int);
extern int send(int, void *, size_t, int);
extern int sendto(int, void *, size_t, int, struct sockaddr *, int);
extern int sendmsg(int, struct msghdr *, int);
extern int setsockopt(int, int, int, void *, int);
extern int shutdown(int, int);
extern int socket(int, int, int);
extern int socketpair(int, int, int, int *);
#endif

```

Problem:

SunOS 4.1.3 defines *SO_DONTLINGER* in *sys/socket.h*. This name is obsolete and is not defined in **FreeBSD 2.0.5**.

Solution:

Use “(~SO_LINGER)” instead of “SO_DONTLINGER”.

2.11 getting timezone information

Problem:

SunOS 4.1.3 provides the timezone name abbreviation in the *struct tm* structure defined in *time.h*. Other systems use the external variable *tzname* in conjunction with the external variable *timezone* to access the timezone name. These external variables are declared in *time.h*.

Solution:

```

char *tz;
...
#ifndef TIMEZONE_IN_STRUCT_TM
    tzset();
    tz = tzname[timezone ? 1 : 0];
#else
    tz = t->tm_zone;
#endif

```

2.12 time.h vs sys/time.h

Problem:

time.h on **SunOS 4.1.3** does not contain all time-related data structures (such as *struct timeval*) and constants some applications may require, nor does it

include *sys/time.h*. Including *sys/time.h* on **SunOS 4.1.3** includes *time.h*. Other systems follow the **Posix** standard where including *time.h* defines all required types and constants.

Solution:

```
#ifndef NEED_SYS_TIME_H
#include <time.h>
#else
#include <sys/time.h>
#endif
```

2.13 tzfile.h and TM_YEAR_BASE

Problem:

FreeBSD 2.0.5, **SunOS**, and **Solaris** all have a *tzfile.h* that contains the line “#define TM_YEAR_BASE 1900”.

HP-UX doesn’t have *tzfile.h* and doesn’t define *TM_YEAR_BASE* in any of the system include files.

Solution:

Since *TM_YEAR_BASE* is intended only to be used with ((*struct tm* *)*t*)->*tm_year* and *tm.year* is the number of years since 1900, instead of “#include <*tzfile.h*>” use

```
#define TM_YEAR_BASE 1900
```

2.14 dynamic library (dl) functions

Problem:

Solaris 2.5, **OSF1 V3.2**, **IRIX 5.3**, **FreeBSD 2.0.5**, and **SunOS 4.1.3**, provide the functions *dlopen()*, *dlsym()*, and *dlclose()* to access objects in dynamic libraries. **HP-UX 9.05**, however, provides the equivalent functions *shl_load()*, *shl_findsym()*, and *shl_unload()*.

Furthermore, **FreeBSD** and **SunOS** do not support any *mode* options on the *dlopen()* call. They require the *mode* argument to be 1.

Solution:

```
#ifndef HAVE_HP_SHLIB           /* not defined == normal system */

#include <dlfcn.h>               /* standard dynamic linker functions */

#else   /* HAVE_HP_SHLIB */        /* naturally HP-UX 9.05 doesn't have the
                                "dl" library all other systems support */

#include <dl.h>

#define RTLD_NOW      BIND_IMMEDIATE
#define RTLD_LAZY     BIND_DEFERRED
```

```

#define dlopen(n, f)      (void *) shl_load((n), (f), 0)
#define dlsym(h, n)       ((shl_findsym((shl_t)(h), (n), TYPE_UNDEFINED,
                                         (void *) &hpsym) == 0) ? hpsym : 0)
#define dlclose(h)        shl_unload((shl_t)(h));
static void *hpsym;
                                         /* makes things non-reentrant, but
                                         HP-UX 9.05 doesn't support threads */

#endif /* HAVE_HP_SHLIB */

#ifndef RTLD_NOW
#define RTLD_NOW           1             /* systems which don't define this expect
                                         the dlopen() mode argument to be 1
*/
#endif /* RTLD_NOW */
#ifndef RTLD_LAZY
#define RTLD_LAZY          1             /* systems which don't define this expect
                                         the dlopen() mode argument to be 1
*/
#endif /* RTLD_LAZY */

```

Note that because most systems do not support the functionality provided by the **Solaris RTLD_GLOBAL** mode option for `dlopen()`, it should not be used.

2.15 MAXHOSTNAMELEN

Problem:

Solaris 2.5 defines the constant `MAXHOSTNAMELEN` in the include file ‘`netdb.h`’, but other systems define it in ‘`sys/param.h`’.

Solution:

Since both files exist on all systems in question, simply include them both.

```
#include <netdb.h> /* define MAXHOSTNAMELEN on Solaris */
#include <sys/param.h> /* define MAXHOSTNAMELEN elsewhere */
```

2.16 Math Errors

Problem:

Programs doing floating-point math, especially those using the STOL expressions (`Sx`) package, need to cleanly handle floating-point exceptions.

On some systems (FreeBSD, for example), operations like divide-by-zero cause a `SIGFPE` to be generated by default. The System V Interface Definition 3 includes the `matherr()` functionality as a way of handling exceptions.

Solution:

No solution to this problem has been recommended as yet.

2.17 Pthreads Types

Problem:

Solaris 2.5 defines pthreads types in the include file ‘`pthreads.h`’, while **Solaris 2.6** and the Single Unix Specification define them in ‘`sys/types.h`’.

The `frame_sorter` program may be compiled with or without threads. When compiled without threads, ‘`pthreads.h`’ is not included and some threads types are defined in the code so it compiles properly. Unfortunately, ‘`sys/types.h`’ is included by many required header files, so on Solaris 2.6, the types are redefined.

Solution:

Don’t redefine pthreads types under any circumstances. Enclose `frame_sorter` code in `#ifdef HAVE_THREADS` and other code in `#ifdef HAVE_PTHREADS` blocks as necessary to avoid redefining pthreads types.

2.18 basename()

Problem:

FreeBSD 3.4 does not include the standard function `basename()`.

Solution:

Add `basename()` to the ‘`configure.in`’ `AC_CHECK_FUNCS` macro and put the following in any code calling `basename()`.

```
#ifndef HAVE_BASENAME
#define basename(s) (strrchr((s), '/') == NULL ? (s) : strrchr((s), '/') + 1)
#endif
```

2.19 socklen_t

Problem:

FreeBSD prior to 4.x and **Solaris** prior to 7 (2.7) do not define the type `socklen_t`.

Solution:

Add ‘`-DSOCKLEN_T=type`’ to the `MY_DEFS` macro in `configure.in` for those systems which do not define that type, where ‘`type`’ is the appropriate type name, such as `int`.

Add the following to source files using `socklen_t`:

```
#ifdef SOCKLEN_T
#define socklen_t SOCKLEN_T
#endif
```

2.20 getaddrinfo()

Problem:

The old functions for looking up names and addresses in the name service return a pointer to a static variable, so they are not re-entrant or thread-safe. They also do not support IPV6. The newer functions, `getaddrinfo()` and `getnameinfo()` are re-entrant and thread-safe and support IPV6, but are not implemented on Solaris 7 and earlier, or HP-UX prior to 11i version 1.6.

Solution:

The ITOS `configure` checks for the presence of `getaddrinfo()`, and uses an alternate implementation from W. Richard Stevens if it is not found. The alternate implementation is '`libunp.a`', which must be installed in `/usr/local/lib` on build systems. Because it is a static library, it need not be installed on machines where ITOS is deployed.

2.21 getopt() and getopt_long()

Problem:

The `getopt_long()` function is not provided by Solaris prior to Solaris 10. Many implementations of `getopt()` and `getopt_long()` provide non-standard extensions.

One of those extensions is that BSD-derived implementations declare `extern int optreset`, which should be set to 1 before using `getopt()` or `getopt_long()` to iterate over `argv` a second or subsequent time.

Solution:

All systems which implement `getopt_long()` seem to have a '`getopt.h`' header file, and some require it to be included, so include it in all cases where `getopt_long()` is provided by the OS.

If `getopt_long()` is not provided by the system, use the BSD implementation which will be included in the ITOS library '`libtcw`' and the '`getopt_long.h`' header file in the ITOS includes directory.

For implementations which provide the global variable `optreset`, define the macro below if you need to iterate over `argv` multiple times, and call the macro between iterations.

```
#define OPTRESET
#if HAVE_GETOPT_LONG
#include <getopt.h>
#else
#include "getopt_long.h" /* local implementation */
#undef OPTRESET
#define OPTRESET (optreset = 1)
#endif
#if HAVE_DECL_OPTRESET
```

```
#undef OPTRESET  
#define OPTRESET (optreset = 1)  
#endif
```

Finally, avoid, or at least be very careful about extensions to the standard which might not be present in all implementations.

Appendix A List of Conditional Variables

D

DECLARES_SOCKET_FUNCS 6

H

HASFINITE 2
HAVE_CLOCK_GETTIME 2
HAVE_HP_SHLIB 8
HAVE_IEEEFP_H 2
HAVE_MALLOC_H 3
HAVE_SYS_MOUNT_H 5
HAVE_SYS_STATVFS_H 5
HAVE_SYS_VFS_H 5

M

MAXHOSTNAMELEN 9

N

NEED_SYS_TIME_H 7
NEED_UNION_SEMUN 3

T

TIMEZONE_IN_STRUCT_TM 7

U

USES_UNION_SEMUN 3

Index

A

accept() 6

B

basename 10
bcopy() 2
bind() 6
bzero() 2

C

connect() 6

D

DECLARES_SOCKET_FUNCS 6
dlopen 8

E

errno 6
exceptions 9

F

FD_SETSIZE 3
finite() 2
floating-point 9

G

getaddrinfo 11
gethostbyaddr 11
gethostbyaddr_r 11
gethostbyname 11
gethostbyname_r 11
getnameinfo 11
 getopt 11
 getopt.h 11
 getopt_long 11
 getpeername() 6
 getprotobynumber 11
 getprotobynumber_r 11
 getsockname() 6
 getsockopt() 6

H

HASFINITE 2
HAVE_CLOCK_GETTIME 2
HAVE_IEEEFP_H 2
HAVE_MALLOC_H 3
HAVE_PTHREADS 10
HAVE_SYS_MOUNT_H 5
HAVE_SYS_STATVFS_H 5
HAVE_SYS_VFS_H 5
HAVE_THREADS 10

I

ieeefp.h 2

K

key_t 3

L

listen() 6

M

malloc.c 3
math 9
MAXHOSTNAMELEN 9
memcpy() 2
memset() 2

N

NEED_SYS_TIME_H 7
NEED_UNION_SEMUN 3
netdb.h 9

P

pthread 10
pthread.h 10

R

recv() 6
recvfrom() 6
recvmsg() 6

S

select()	3
semaphore	3
semctl	3
send()	6
sendmsg()	6
sendto()	6
setsockopt()	6
shutdown()	6
SIGFPE	9
SO_DONTLINGER	6
socket()	6
socketpair()	6
socklen_t	10
statfs	5
statvfs	5
strerror	6
struct timespec	2
struct timeval	2
struct tm	7
sys/mount.h	5
sys/param.h	9
sys/socket.h	6

sys/statvfs.h	5
sys/time.h	7
sys/types.h	10
sys/vfs.h	5
sys_errlist	6
sys_nerr	6

T

time	2, 7
time.h	7
timezone	7
TIMEZONE_IN_STRUCT_TM	7
TM_YEAR_BASE	8
tzfile.h	8
tzname	7
tzset()	7

U

ulimit()	3
union semun	3
USES_UNION_SEMUN	3

Table of Contents

1	Introduction	1
2	Portability issues	2
2.1	bcopy() vs memcpy() (and bzero() vs memset())	2
2.2	clock_gettime() vs gettimeofday()	2
2.3	finite()	2
2.4	key_t	3
2.5	malloc.h	3
2.6	select() and ulimit() vs. FD_SETSIZE	3
2.7	semctl() and union semun	3
2.8	statfs vs statvfs	5
2.9	sys_errlist and sys_nerr	6
2.10	sys/socket.h	6
2.11	getting timezone information	7
2.12	time.h vs sys/time.h	7
2.13	tzfile.h and TM_YEAR_BASE	8
2.14	dynamic library (dl) functions	8
2.15	MAXHOSTNAMELEN	9
2.16	Math Errors	9
2.17	Pthreads Types	10
2.18	basename()	10
2.19	socklen_t	10
2.20	getaddrinfo()	11
2.21	getopt() and getopt_long()	11
	Appendix A List of Conditional Variables	13
	Index	14